

TP 5 - éléments de correction

```
1 import numpy as np
2 import matplotlib.pyplot as plt
3
4 # Fonctions pour les tests
5 def P_0(x):
6     return np.ones_like(x)
7 def P_1(x):
8     return x
9 def P_2(x):
10    return x**2
11 def P_3(x):
12    return x**3
13 def P_4(x):
14    return x**4
15 def P_5(x):
16    return x**5
17 def inv(x):
18    return 1/x
19
20 # Routine rectangles à gauche
21 def rect1(f,a,b,n):
22    x = np.linspace(a,b,n+1)
23    s = 0
24    for k in range(0,n):
25        s += f(x[k])
26    return (b-a)/n * s
27
28 def rect2(f,a,b,n):
29    x = np.linspace(a,b,n+1)
30    s = np.sum(f(x[0:n]))
31    return (b-a)/n * s
32
33 # Routine point du milieu
34 def ptmilieu1(f,a,b,n):
35    x = np.linspace(a,b,n+1)
36    z = np.zeros(n)
37    for k in range(n):
38        z[k] = 0.5 * (x[k] + x[k+1])
39    s = 0
40    for k in range(n):
41        s += f(z[k])
42    return (b-a)/n * s
43
44 def ptmilieu2(f,a,b,n):
45    x = np.linspace(a,b,n+1)
46    z = (x[0:n] + x[1:n+1])/2
47    s = np.sum(f(z))
48    return (b-a)/n * s
49
50 # Routine trapèze
51 def trap1(f,a,b,n):
52    x = np.linspace(a,b,n+1)
53    s = 0
54    for k in range(1,n):
55        s += f(x[k])
56    return (b-a)/(2*n) * (f(a) + 2*s + f(b))
57
58 def trap2(f,a,b,n):
59    x = np.linspace(a,b,n+1)
60    s = np.sum(f(x[1:n]))
61    return (b-a)/(2*n) * (f(a) + 2*s + f(b))
62
63 # Routine Simpson
64 def simpson1(f,a,b,n):
65    h = (b-a)/n
66    x = np.linspace(a,b,n+1)
67    s1 = np.sum(f(x[1:n]))
68    xkp12 = (x[0:n] + x[1:n+1])/2
69    s2 = np.sum(f(xkp12))
70    return h/6 * (f(a) + f(b) + 2*s1 + 4*s2)
```

```

71
72 def simpson2(f,a,b,n):
73     h = (b-a)/n
74     x = np.linspace(a,b,n+1)
75     s1 = np.sum(f(x[1:n]))
76     xkp12 = (x[0:n] + x[1:n+1])/2
77     s2 = np.sum(f(xkp12))
78     return h/6 * (f(a) + f(b) + 2*s1 + 4*s2)
79
80 # Erreurs pour tracé
81 def erreur_rect2(f,vec_n):
82     return np.array([abs(rect2(f,1,2,2*n) - np.log(2)) for n in vec_n])
83
84 def erreur_ptmilieu2(f,vec_n):
85     return np.array([abs(ptmilieu2(f,1,2,2*n) - np.log(2)) for n in vec_n])
86
87 def erreur_trap2(f,vec_n):
88     return np.array([abs(trap2(f,1,2,2*n) - np.log(2)) for n in vec_n])
89
90 def erreur_simpson2(f,vec_n):
91     return np.array([abs(simpson2(f,1,2,n) - np.log(2)) for n in vec_n])
92
93 # Fonctions de référence
94 def inv_2(x):
95     return 1/x**2
96 def inv_3(x):
97     return 1/x**3
98 def inv_4(x):
99     return 1/x**4
100
101 # Tracé des erreurs
102 vec_n = np.arange(10,1001,10)
103 f = inv
104
105 plt.loglog(vec_n, erreur_rect2(f,vec_n), 'x', label="erreur_rect2")
106 plt.loglog(vec_n, erreur_ptmilieu2(f,vec_n), 'o', label="erreur_ptmilieu2")
107 plt.loglog(vec_n, erreur_trap2(f,vec_n), '+', label="erreur_trap2")
108 plt.loglog(vec_n, erreur_simpson2(f,vec_n), '.', label="erreur_simpson2")
109
110 vec_x = np.linspace(10,1001,100000)
111 plt.loglog(vec_x, inv(vec_x), label="1/x")
112 plt.loglog(vec_x, inv_2(vec_x), label="1/x^2")
113 plt.loglog(vec_x, inv_3(vec_x), label="1/x^3")
114 plt.loglog(vec_x, inv_4(vec_x), label="1/x^4")
115
116 plt.legend()
117 plt.show()

```
